INCREASING ACCURACY THROUGH CLASS DETECTION: ENSEMBLE CREATION USING OPTIMIZED BINARY KNN CLASSIFIERS

Benjamin Thirey[1] and Christopher Eastburg[2]

[1]Department of Mathematical Sciences, United States Military Academy, West Point, New York, USA
benjamin.thirey@usma.edu
[2]Department of Mathematical Sciences, United States Military Academy, West Point, New York, USA
christopher.eastburg@usma.edu

ABSTRACT

Classifier ensembles have been used successfully to improve accuracy rates of the underlying classification mechanisms. Through the use of aggregated classifications, it becomes possible to achieve lower error rates in classification than by using a single classifier instance. Ensembles are most often used with collections of decision trees or neural networks owing to their higher rates of error when used individually. In this paper, we will consider a unique implementation of a classifier ensemble which utilizes kNN classifiers. Each classifier is tailored to detecting membership in a specific class using a best subset selection process for variables. This provides the diversity needed to successfully implement an ensemble. An aggregating mechanism for determining the final classification from the ensemble is presented and tested against several well known datasets.

KEYWORDS

k Nearest Neighbor, Classifier Ensembles, Forward Subset Selection

1. INTRODUCTION

1.1. k-Nearest Neighbor Algorithm

The k-Nearest Neighbors, or kNN algorithm is well known to the data mining community, and is one of the top algorithms in the field [1]. The algorithm achieves classification between m different classes. Each instance to be classified is an item that contains a collection of r different attributes in set $A=\{a_1, a_2, ..., a_r\}$ where a_j corresponds to the j^{th} attribute. Therefore, an instance is a vector $\mathbf{p} = \langle p_1, p_2, \ldots, p_r \rangle$ of attribute values. For some predetermined value of k, the nearest k neighbors are determined through the use of a distance metric which is calculated using the difference in distances between each of the attributes of the instance in question and its neighbors. Euclidean distance is by far the most popular metric for calculating proximity. An instance's membership within a given class can be computed either as a probability or by simple majority of the class with the most representation in the closest k neighbors. At the simplest level, this is a problem of binary classification, where data is classified as being in a certain class or not.

Due to different units of measurement, there is also a need for normalization across attribute variables in order to prevent one variable from dominating the classification mechanism [2]. One of the problems with kNN is that without some sort of weighting scheme for variables, each of the variables is treated as being equally important toward determining similarity

DOI : 10.5121/ijcsea.2011.1201

between instances. Combining different scales of measurement across attributes when computing the distance metric between instances can cause severe distortions in the calculations for determining nearest neighbors. Several different variable weighting schemes and selection methods to overcome this are discussed by Wettschereck, Aha, Mohri [3]. Given the means by which neighbors in kNN are calculated, irrelevant variables can have a large effect on final classification. This becomes especially problematic in cases where a large number of predictor variables are present [4]. Closely related to this problem is the curse of dimensionality whereby the average distance between points becomes larger as the number of predictor variables increases. One of the benefits of proper variable selection is that it has the potential to help mitigate the curse of dimensionality.

It is generally held that kNN implementations are sensitive to the selection of variables, so choice of the appropriate subset of variables for use in classification plays a critical role [5]. One of the methods is through the use of forward subset selection (FSS) with the kNN algorithm [6]. FSS begins by identifying the variable which leads to the highest amount of accuracy with regards to classifying an instance. That attribute is then selected for inclusion in the subset of best variables. The remaining variables are then paired up with the set, and the next variable for inclusion is again calculated by determining which one leads to the greatest increase in classifier accuracy. This process of variable inclusion continues until no further gains can be made in accuracy. Clearly, this is a greedy method of determining attributes for inclusion since the variable selected at each step is the one providing the biggest gains in accuracy. Therefore, the subset selected at the conclusion of the algorithm will not necessarily be the most optimal since not all potential combinations of variables were considered. Additionally, this algorithm is quite processor intensive.

Backward subset selection (BSS) operates in a similar manner, except that all variables are initially included and then a variable is discarded during each pass through the attributes until no further improvements in accuracy are achieved. Work by Aha and Bankert [6] found that FSS of variables led to higher classification rates than BSS. They also conjectured that BSS does not perform as well with large numbers of variables.

kNN relies on forming a classification based on clusters of data points. There are a variety of ways to consider kNN clusters for final classification. Simple majority rule is the most common, but there are other ways of weighting the data [1]. Wettshereck, Aha, and Mohri [5] provide a comprehensive overview of various selection and weighting schemes used in lazy learning algorithms, such as kNN, where computation is postponed until classification. These modifications to the weighting calculations of the algorithm include not only global settings, but local adjustments to the weights of individual variables. The weights are adjustable depending on the composition of the underlying data. This allows for greater accuracy and adaptability in certain portions of the data without imposing global variable weightings.

1.2. Classifier Ensembles

Classifier ensembles render classifications using the collective output of several different machine learning algorithms instead of only one. Much of the initial development of ensemble methods came through the use of trees and neural nets to perform classification tasks. It was recognized that the use of aggregated output from multiple trees or neural nets could achieve lower error rates than the classification from a single instance of a classifier. The majority of the research in the area of ensembles uses either decision tree or neural net classifiers. Work regarding ensemble selection from a collection of various classifiers of different types has been successful in generating ensembles with higher rates of classification [7].

There are a number of methods for generating classifiers in the ensemble. In order to be effective, there must be diversity between each of the classifiers. This is usually achieved through the utilization of an element of randomness when constructing the various classifiers. According to the distinctions made by Brown, et al. [8] diversity can be either explicit through

the use of deterministic selection of the variables with individual classifiers, or it can be implicit since diversity is randomly generated. For example, implicit methods achieve diversity through the initialization of the weights of a neural net at random or using a randomized subset of features when node splitting in trees.

The development of individual classifiers for use by decision tree or neural net ensembles is usually performed with a random subset of predictor variables. This is to provide diversity and ensure that errors are more likely to occur in different areas of the data. This process is repeated numerous times so that a wide variety of classifiers is produced, and the necessary diversity amongst individual classifiers is established. Recent research compares how the various means of generating classifiers compares with the output of their respective ensembles [9]. Techniques such as bagging and boosting are used to generate different classifiers that make independent classifications of instances. Bagging is a technique where the underlying dataset is repeatedly sampled during the training phase, whereas boosting changes the distribution of the training data by focusing on those items which present difficulties in classification [10].

Researchers examined the use of kNN classifiers as members of an ensemble [11]. Madabhushi, et al. found that using kNN ensembles on Prostate Cancer datasets resulted in higher accuracy rates than other methods which required extensive training [12]. Work by Bay considered an ensemble of kNN classifiers which were developed from random subsets of variables [13]. This method resulted in increased classification accuracy. The objective of developing these different classifiers is to ensure that their respective errors in classification occur in different clusters of data. Domeniconi and Yan [14] proposed a method whereby different subsets of variables were randomly generated and used to construct members of kNN ensembles. Their approach continued by adding only those classifiers to the ensemble which improved ensemble classification performance.

Use of the classifier ensemble is straightforward. Consider an ensemble $C^* = \{c_1, c_2, \ldots, c_m\}$ of *m* individual classifiers, with each as a binary classifier. The instance to be classified is passed through the group of classifiers C* and their corresponding individual classifications are then aggregated as discussed above in order to determine what the final classification should be.

The final step in developing an ensemble classifier is to determine how each of the votes from the individual classifiers in the ensemble will be transformed into a final classification. The most common method is to use a simple majority rule, but it is not difficult to see how various weighting schemes could be implemented in this process. Perhaps the occurrence of the classification as membership of a particular class is enough to override all other votes. The underlying data and application are the primary decision criteria regarding how votes should be tallied.

1.3. Related and Recent Work

The fundamental strategy of ensemble network classification is to generally isolate errors within different segments of a population. Oliveira et al, [15], used genetic algorithms to generate ensembles for classification models of handwriting recognition. Their methodology uses genetic programming to continually create new networks, search for the best features, and keep the set of networks that are both accurate but disagree with each other as much as possible. Error rates in final classification will be less when ensembles use only a subset of the best features for classification. A supervised and an unsupervised approach were used to extract the best features relevant for subset selection and ensemble creation. They found that both techniques were successful and also concluded there are still many open problems with regard to optimal feature extraction.

K means clustering is a popular classification clustering algorithm by where each observation is a member of the cluster with the nearest mean. K medoids is a similar approach which uses actual data points for cluster centers. [2] K means does not work well with data clusters which

are non spherical and of different sizes. There are many techniques in literature to improve the k means algorithm. For example, fuzzy k means clustering often improves results by incorporating a probabilistic component into membership classification. Weng et al [16] effectively used ensembles of k means clustering to improve the classification rate of intrusion detection for computer network security. Their approach successfully improves classification with clusters of "anomalistic shapes." Work by Bharti et al [17] used a decision tree algorithm, known as J48, built with fuzzy K-means clustering to very accurately map clusters of data to classification for intrusion detection.

Awad et al [18] recently applied six different machine learning algorithms to spam classification: Naïve Bayes, Support Vector Machines (SVM), Neural Networks, k-Nearest Neighbor, Rough Sets, and the Artificial Immune System. While performing well in the categories of spam recall and overall accuracy, kNN showed a marked decrease in precision (the ability to correctly filter out noise) compared to the other algorithms. Used here, the kNN routine produced too many false positives. Perhaps using an ensemble of kNN classifiers would have significantly improved results.

It is recognized that kNN is very sensitive to outliers and noise within observations. Jiang and Zhou, [19] built four kNN classification techniques involving editing and ensemble creation. In order to manage the error induced by outliers, they developed differing editing routines that effectively removed the most problematic training data and therefore increased the accuracy of classification. They also created a fourth neural network ensemble mechanism using the Bagging technique, which generally performed better than the editing routines. An approach used by Subbulakshmi et al [20] also used several different classifier types (neural nets and SVMs) to enhance overall classification. Each of the individual classifiers of the ensemble possessed different threshold values for activation based on the ensemble member's accuracy. They found that the ensemble approach had higher classification rates than any of the individual underlying classifiers.

2. OUR APPROACH

Our approach begins with the production of an ensemble of kNN classifiers. We chose to use kNN classifiers because of their ability to adapt to highly nonlinear data, they are a fairly mature technique, and there are a number of methods available for optimizing instances of kNN classifiers.

Each instance or object to be classified $\boldsymbol{p}$ is a vector of values for r different attributes so $\boldsymbol{p} = \langle p_1, p_2, \dots, p_r \rangle$. We follow the one-per-class method of decomposing classification amongst m different classes into a set of binary classification problems [21]. These classifiers determine class membership using a set $\boldsymbol{C}^* = \{c_1, c_2, \dots, c_m\}$, where classifier $c_i \in \boldsymbol{C}^*$ determines if a given instance is member of the i^{th} class. Each classifier takes a vector of attributes for an item to be classified and performs the following function:

$$c_i(\boldsymbol{p}) = \begin{cases} 1 \; if \; \boldsymbol{p} \; is \; an \; element \; of \; class \; i \\ 0 \; if \; \boldsymbol{p} \; is \; NOT \; an \; element \; of \; class \; i \end{cases}$$

This method works best for algorithms such as kNN that produces activation as an output to determine class membership [22]. Essentially each binary kNN classifier is the analogue of a classification "stump", which is a decision tree that makes a single classification of whether or not a given instance is a member of a specific class. Classifiers which discriminate between all classes, such as a single model to determine membership, have an error rate determined by the number of misclassifications from the entire dataset. This is because the classifier is tailored for and optimized over the collection of m different classes. As a result, the parameters are adjusted

so that the error rate across all classifications is as low as possible without deference to any particular class.

The subset of variables which leads to the lowest error rates when determining membership in a specific class are likely to be entirely different from the subset of variables which are most effective in determining membership in another class. The use of the FSS algorithm allows each individual binary classifier to tailor itself around the variables it deems most important for determining membership of an instance. As a result, diversity amongst the kNN classifiers is achieved deterministically. The necessary diversity is achieved by each individual classifier selecting the set of variables which are deemed most important for identifying specific class membership. This is slightly different from the traditional definition of diversity which stresses errors being made on different instances of data.

Since we use an ensemble of individual kNN classifiers which are responsible for determining membership in a specific class, each individual classifier can have the parameters for variable weights adjusted to achieve the highest classification rate for the specific class being analyzed. When using a single classifier to differentiate between multiple classes, the differences in which variables are most important to clustering for identification of various classes becomes overshadowed.

Our approach also differs from previous approaches in that we use the specialized kNN technique of FSS for each of the binary classifiers in the ensemble. We elected to use FSS since the final collection of predictor variables selected for classification is usually smaller [6]. This is especially noticeable in datasets with a large number of variables for subset selection. We also chose FSS as opposed to BSS since it requires significantly less processor time, especially given the large amount of processing time which must be devoted if there are many variables. Furthermore, the models are often substantially simpler. As discussed previously, a successful ensemble implementation requires diversity between the individual classifiers being used. The diversity here is achieved through the inclusion of different variables which are selected by the FSS-kNN algorithm as being the most important towards determining membership in an instance of a particular class.

By building different classifiers for determining membership in each class, we are choosing the subset of variables that work best with the kNN algorithm to classify members of the specific class. This provides the algorithm with greater accuracy than a single implementation of the kNN algorithm differentiating amongst all classes. Through an individual classifier tailored to determine membership for a particular class, we allow the isolation of those variables that contribute the most toward the clustering of the class members. Clearly, the subsets of variables selected between different binary classifiers will be different. Furthermore, by using a kNN variant collection which has been optimized, the ensemble itself should have a higher resultant classification rate. There are many other implementations of the kNN process that we could have relied on. We believe that the use of any of these others would lead to similar results.

One of the benefits of this method is that it overcomes the curse of dimensionality. For a given class, there might only be a handful of variables which are critical to classification and could be completely different from other classes. A classifier differentiating between all m classes would have to potentially consider all attributes. However, our approach relies on the fact that the classifier of the i^{th} class needs only to determine membership through the use of variables that are most important to determining distance to its nearest neighbors.

In order to combine the respective votes of each member within the ensemble, we have three cases: one of the individual classifiers identifies membership within the group, no membership is selected, or there is a conflict regarding classification with two or more classifiers presenting conflicting classifications. Where classification is straightforward with one classification emerging from the ensemble, we use that classification. In the latter two cases discussed above, there must be some way of achieving an output. There are two possible approaches. The first is to rely on a single overall kNN classifier which determines identification in the event of

conflict. Therefore, if the ensemble is unsuccessful, the classification scheme reverts back to a single instance (*master*) classifier. The second approach is to use the classifier with the highest accuracy which selected the instance for membership. Figure 1 presents an overview of this process.

A *master* classifier uses the same methodology but provides for classification between all possible classes in the dataset as opposed to simply determining membership in a single class. This master classifier is used to assign classification in the event that none of the members of the ensemble identifies an item for class membership.

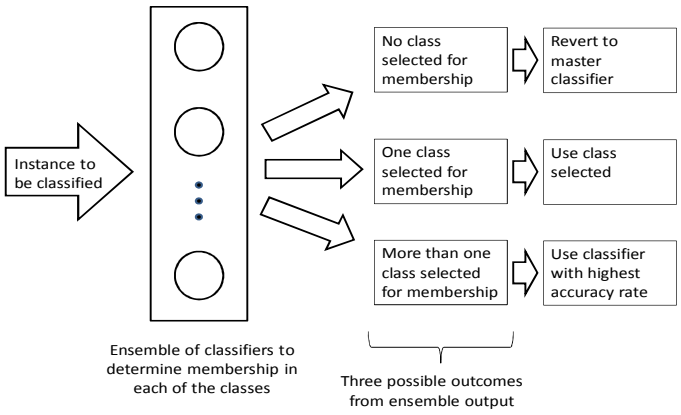

Figure 1. Determining class membership of an instance

3. EXPERIMENTAL RESULTS

3.1. Datasets

The datasets which we utilized were from the UCI Machine Learning Repository with the exception of the IRIS dataset which is available in the R software package [23, 24, 25]. The statistics regarding each data set are presented in Table 1. We began with the IRIS data since it is one of the most used datasets in classification problems. Furthermore, it is a straightforward dataset with four predictors and provided a good benchmark for initial results. We also selected the Low Resolution Spectrometer (LRS) data since it contained a large number of variables and the data required no scaling prior to using the algorithm. The dataset itself consists of header information for each entry, followed by intensity measurements at various spectral lengths. Finally, the ARRYTHMIA dataset was selected due to the large number of predictor variables which it offered. We were curious to see how well the FSS-kNN algorithm performed at reducing the number of variables needed to determine class membership. There were several instances in the ARRYTHMIA data set where missing data was problematic. These attributes were removed from the dataset so that classification could continue.

Table 1. Dataset Statistics

Dataset:	**Iris**	**LRS**	**Arrythmia**
Number of classes:	3	10	16
Number of variables:	4	93	263
Number of data points:	150	532	442

3.2. Model Generation

Our methodology follows the pseudocode in Figure 2. We began by building the best classification model for each class in the dataset. The individual models were constructed using FSS-kNN to determine the best subset of variables to use for determining membership in each class. Every subset of variables was then tested using *n*-fold cross validation, where each element was predicted using the remaining $n-1$ elements in the kNN model over various k-values to determine the most accurate models for each class. This required a modest amount of processor time, but enabled us to use all of the available data for both training and testing which is one of the benefits of *n*-fold cross validation. Following the generation of the individual classifiers, we built the master classifier.

After building our classifiers, we processed the data with the ensemble. The majority of instances were selected for membership by one of the classifiers. In the event that more than one classifier categorized the instance as being a member of the class that it represented, we reverted to the model accuracies of the individual classifiers, and assigned the item to the most accurate classifier which identified the item for class membership. Instances which were not selected for membership in a class by any of the individual classifiers were processed by the master classifier.

We conducted *n*-fold cross-validation testing to determine the overall accuracy of the ensemble. The *k*-value and subset of variables selected for an individual kNN classification model were the only factors remaining the same between the classifications of instances.

```
CONSTRUCTION PHASE:
for each class in the data set
   build classifier c_i which determines membership in class i using the Forward Subset Selection
Algorithm
   compute the accuracy of this classifier
next class

build a master classifier which considers membership amongst all classes

CLASSIFICATION PHASE:
for each item to be classified
   the item is evaluated by each classifier so seek membership in respective classes
   if only one classifier identified the item for membership
     then assign the item to that class
  if more than one classifier identified the item for membership
     then assign class membership to the most accurate classifier
  if no classifiers identified the item for membership
     then use the master classifier to assign a classification
next item
```

Figure 2. Pseudocode for classifier construction and usage

4. DISCUSSION OF RESULTS

The statistics regarding the accuracy rates and numbers of predictor variables used by the individual classifiers are presented in Table 2. By using individual classifiers to determine set inclusion, we were able to achieve high rates of classification. Only predictor variables useful in the classification of instances of a given class with the kNN algorithm were used in the models. In the LRS and ARRYTHMIA datasets, the largest model for membership classification in the ensemble uses only about 5% of the available predictor variables. The average number of predictor variables used for classification is significantly less than that. In each of the datasets, there were classification models that needed only one variable with which to determine membership in a particular class. We believe that the high rates of classification

for the individual classifiers are closely related to the reduction in the number of dimensions being used for model construction, thereby overcoming the curse of dimensionality. This has some rather interesting implications. The first is that this process can be used as a discovery mechanism for determining the most important variables for use in determining membership in a specific class. It also implies that accurate models can be constructed using small subsets of available predictor variables, thereby greatly reducing the number of dimensions in which classifications are performed.

The results of building the master models that incorporate all of the classes are depicted in Table 2. These represent use of a single model constructed using the FSS-kNN algorithm to determine classifications of data. Note that the classification accuracy rates of the models that determine classification between all classes are at or below the minimum accuracy rates of the individual classifiers that determine membership in a specific class. This is not surprising given that the master classifier is now attempting to discriminate between all of the various classes. Note also that the numbers of variables selected by the master models are significantly higher than the mean number of variables selected in the individual classifiers. These represent the best accuracy rates available if only one classifier was being constructed using the FSS-kNN model.

Table 2. FSS-kNN Statistics for Classifiers of the Ensemble

Dataset:	**Iris**	**LRS**	**Arrythmia**
Max Accuracy Achieved:	1.000	0.998	1.000
Mean Accuracy of Classifiers in Ensemble:	0.979	0.985	0.977
Standard Deviation of Classifiers in Ensemble:	0.018	0.015	0.043
Minimum Accuracy of Classifiers in Ensemble:	0.973	0.957	0.827
Maximum Variables Selected by a Classifier:	2	5	13
Mean Number of Variables Selected by Classifiers in Ensemble:	1.667	2.900	2.750
Standard Deviation of Number of Variables Selected:	0.577	1.524	3.066
Minimum Variables Selected by a Classifier	1	1	1
Accuracy Rate of the Master Model:	0.973	0.891	0.732
Number of Variables Utilized in Master Model:	2	6	6

When classifying instances, there were three distinct cases that could occur. An instance could be selected for membership by none, one, or more than one of the ensemble classifiers. Table 3 presents the statistics for various parts of our method.

The accuracy of the master classifier used in cases where no ensemble classifier identified membership demonstrated a significant degradation in classification accuracy. This probably represents classification of difficult cases which were largely responsible for the errors in the master classification. Instances which are not selected for classification by any of the individual ensemble members are passed to the master classifier. This classifier is hindered by the same difficulties that individual classifiers face when determining membership of an object in a specific class. Here though, we are forcing a classification to take place.

Table 3. Ensemble Statistics and Accuracy Rates

Dataset:	Iris	LRS	Arrythmia
Instances Classified by 0 Members of the Ensemble:	0 (0%)	34 (6.4%)	56 (12.42%)
Accuracy of Master Model in Determining Class Membership:	N/A	0.4705882	0.4464286
Instances Classified by 1 Member of Ensemble:	145 (96.66%)	483 (90.96%)	375 (83.15%)
Accuracy:	0.9862068	0.9689441	0.8773333
Instances Classified by 2 Members of the Ensemble:	5 (3.33%)	14 (2.63%)	20 (4.43%)
Accuracy:	0.8	0.7142857	0.8
Instances Classified by 3 Members of the Ensemble:	0	0	0
Overall Accuracy of Method:	**0.98**	**0.930255**	**0.820399**

Instances classified by only one of the ensemble members comprised the majority of cases in classification and were characterized by their large degree of accuracy. Instances selected for membership in a class by two or more of the ensemble members comprised a small minority of classification cases. By reverting to classifier accuracy to determine the final classification, we were able to achieve fairly high classification rates, given that blind chance would have resulted in a 50% accuracy rate. There were no cases in any of our data sets where more than two classifiers competed for a given instance.

The overall accuracy obtained by the ensemble method presented in this paper is greater than the single classifier attempting to classify amongst all classes. Consequently, using ensembles increases accuracy when compared to the case of using a single classifier.

5. CONCLUSIONS AND FUTURE WORK

Our approach has demonstrated that an ensemble of classifiers trained to detect membership in a given class can achieve high rates of classification. We have shown that we can achieve greater classification rates by combining a series of classifiers optimized to detect class membership, than by using single instances of classifiers. Our model is best adapted towards classification problems involving three or more classes since a two class model can be readily handled by a single classifier instance.

We have not adjusted the importance of individual variables during the process of constructing individual classifiers for the ensemble. We have simply included or excluded variables as being equally weighted without scaling. While variable selection is helpful in addressing some of the problems outlined, additional improvements can be made to the kNN algorithm by weighting the variables which have been selected for inclusion into the model to account for differences in variable importance. Another weakness which needs to be addressed is the consideration of incomplete datasets.

Future work will focus on developing additional classifiers to distinguish between instances that are selected for class membership by more than one classifier within the ensemble rather than reverting to the highest accuracy rate. An element of conditional probability might be of

considerable importance in biomedical classifications. In larger datasets, there could be a number of cases where discerning membership amongst instances becomes difficult. Often the determination occurs between two classes which are very similar. In such cases where FSS-KNN results in classifiers with relatively low rates of classification, it might be necessary to examine the data to determine whether the class in question is really composed of several subclasses which would benefit from their own respective binary classifiers within the ensemble. Finally, there remains the possibility that we can use the predictor variables selected as most important for clustering by FSS to improve classification rates of other methods such as neural nets and decision trees.

REFERENCES

[1] Xindong, W., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Yu, P. S., Zhou, Z. H., Steinbach, M., Hand, D. J. and Steinberg, D. (2008) Top 10 Algorithms in Data Mining. Knowledge and Information Systems, 14 1: 1-37.

[2] Xu, R., Wunsch, D., "Clustering", 2009, John Wiley & Sons

[3] Wettschereck, D., Aha, D., & Mohri, T. (1995). A Review and Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms. Tech. rept. AIC95-012. Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence, Washington, D.C.

[4] Hand, D. , Mannila, H., Smyth, P., Principles of Data Mining, 2001

[5] Kotsiantis, S. "Supervised machine learning: a review of classification techniques", Informatica Journal (31), 2007, pp.249-268.

[6] Aha, D. W., & Bankert, R. L. (1996). A Comparative Evaluation of Sequential Feature Selection Algorithms. In D. Fisher & H. H. Lenz (Eds.), Artificial Intelligence and Statistics V. New York: Springer – Verlag.

[7] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes. "Ensemble selection from libraries of models." in Proceedings of the International Conference on Machine Learning (ICML), 2004.

[8] G. Brown, J. Wyatt, R. Harris, and X. Yao. Diversity creation methods: A survey and categorisation. Journal of Information Fusion, 6(1):5–20, 2005.

[9] D. Opitz, R. Maclin, Popular ensemble methods: An empirical study, Journal of Articial Intelligence Research 11 (1999), pp 169 - 198.

[10] Tan, Pang-Ning, Steinbach, Michael, Kumar, Vipin, Introduction to Data Mining, Addison Wesley, 2006, pp 283-285.

[11] N. El Gayar, An Experimental Study of a Self-Supervised Classifier Ensemble, International Journal of Information Technology, Vol. 1, No. 1, 2004.

[12] A. Madabhushi, J. Shi, M.D. Feldman, M. Rosen, and J. Tomaszewski, "Comparing Ensembles of Learners: Detecting Prostate Cancer from High Resolution MRI," Proc. Second Int'l Workshop Computer Vision Approaches to Medical Image Analysis (CVAMIA '06), pp. 25-36, 2006.

[13] S. D. Bay. Combining Nearest Neighbor Classifiers Through Multiple Feature Subsets. Proc. 17th Intl. Conference on Machine Learning, pp. 37 – 45, Madison, WI, 1998.

[14] C. Domeniconi and B. Yan. Nearest Neighbor Ensemble. In Proceedings of the 17th International Conference on Pattern Recognition, Cambridge, UK, pages 23–26, 2004.

[15] L. S. Oliveira, M. Morita, R. Sabourin, and F. Bortolozzi. Multi-Objective Genetic Algoritms to Create Ansemble of Classifiers. Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization, Vol. 87, pp 592-606. 2005.

[16] Fangfei Weng, Quingshan Jiang, Liang Shi, and Nannan Wu., An Intrusion Detection System Based on the Clustering Ensemble, IEEE International Workshop on 16-18 April 2007, pp. 121-124. 2007.

[17] K. Bharti, S. Jain, S. Shukla, Fuzzy K-mean Clustering Via J48 For Intrusion Detection System. International Journal of Computer Science and Information Technologies, Vol 1(4), pp 315-318, 2010.

[18] W. A. Awad and S. M. ELseuofi, Machine Learning Methods for Spam E-Mail Classification, International Journal of Computer Science & Information Technology, Vol 3, No 1, pp 173 – 184. 2011.

[19] Y. Jiang and Z-H. Zhou. Editing Training Data for kNN Classifiers with Neural Network Ensemble. Lecture Notes in Computer Science 3173, pp 356 – 361, 2004.

[20] T. Subbulakshmi, A. Ramamoorthi, and S. M. Shalinie, Ensemble Design for Intrusion Detection Systems. International Journal of Computer Science & Information Technology, Vol. 1, No. 1, August 2009, pp 1 – 9.

[21] Nilsson, Nils J., Learning Machines: Foundations of Trainable Pattern-Classifying Systems., McGraw-Hill, 1965

[22] Kong, E. B., & Dietterich, T. G., Error-Correcting Output Coding Corrects Bias and Variance, in A. Prieditis & S. Russell, eds, Machine Learning: Proceedings of the Twelfth International Conference, San Francisco, CA: Morgan Kaufmann, pp. 313 – 321. 1995.

[23] Frank, A. & Asuncion, A. (2010). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml/datasets/Arrhythmia]. Irvine, CA: University of California, School of Information and Computer Science.

[24] Frank, A. & Asuncion, A. (2010). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml/datasets/Low+Resolution+Spectrometer]. Irvine, CA: University of California, School of Information and Computer Science.

[25] R: A Language and Environment for Statistical Computing, R Development Core Team, Vienna, Austria, Version 2.11.1.